How to Host a Cydia™ Repository

A few months ago, I introduced Cydia and Telesphoreo. Since then, I have received a number of questions on how to create a Cydia "source". Many people assume that Cydia uses the same repository structure as AppTapp Installer, which is definitely not the case as then it would solve none of AppTapp's packaging issues. Luckily, the answer to this question is usually quite short: a Cydia repository *is* a Debian APT repository.

However, even though Debian APT is quite well documented, this hasn't quite enough for some people, partly due to all of the diversions the official documentation normally takes on its route to a repository. Also, Cydia *does* have a few extensions to the underlying structure that are worth documenting. I have therefore put together this document to help guide people into the world of distributing their software using Cydia using the simplest possible infrastructure.

Step 1: Making a Package

In the world of Debian APT/dpkg, anything you can install is a "package". These packages exist in the form of .deb files, which have a rather arcane (which, to the software historian, may be read as "fascinating") internal format that this guide will not bore the reader with. At a high level, a .deb contains the files that are going to installed as part of the package, and "control" information which comprises metadata about the package (its name, size, and other sundry details).

The construction of such a file is done using the tool dpkg-deb. Users of Debian (or Fink on a Macintosh, which is also based on APT) will already have this package installed, but users of almost any other platform are able to install it easily, or may do this work on their iPhone itself. In order to speed this process along for the recalcitrant few, I have compiled statically linked copies for Linux and Mac OS X 10.4+.

To then make a package, we need only prepare a folder that contains the files we want to install as they would appear on the iPhone's filesystem. Additionally, we will add a single directory called DEBIAN to the root of our package to house a file called "control", which will contain our metadata. If, for example, we want to install a program and a LaunchDaemon for MyProgram, we (depending on what we were installing, of course) may end up with the following directory structure:

```
+- MyProgram
   +- Applications
   |  +- MyProgram.app
   |     +- Info.plist
   |     +- MyProgram
   |     +- icon.png
   +- DEBIAN
   |  +- control
   +- System
      +- Library
         +- LaunchDaemons
            +- com.saurik.MyProgram.plist
```

The contents of the control file is a series of name/value pairs (separated by a colon), one per line. Here is an example of such a completed control file, along with a broken-out description of what each of the fields mean and the types of values they contain.

```
Package: com.saurik.myprogram
Name: MyProgram
Version: 1.0.4-1
Architecture: iphoneos-arm
Description: an example of using APT
 Every day people use Cydia, but with the
 instructions embodied in this package,
 people can publish for it as well.
Homepage: http://www.saurik.com/id/7
Depiction: http://www.saurik.com/id/7
Maintainer: Your Name <you@example.com>
Author: Jay Freeman (saurik) <saurik@saurik.com>
Sponsor: Microsoft <http://www.microsoft.com/>
Section: Games
```

- *Package*: This is the "identifier" of the package. This should be, entirely in lower case, a reversed hostname (much like a "bundleIdentifier" in Apple's Info.plist

Jay Freeman (saurik)
Technology Consultant
saurik@saurik.com
(805) 895-7209

Follow me on Twitter!

The College of Creative Studies

Recent Articles

Yet Another Android Master Key Bug
Android Bug Superior to Master Key
Exploit (& Fix) Android "Master Key"
Exploiting a Bug in Google's Glass
Where did my iOS 6 TSS data go?
When "Dumb Pipes" Get Too Smart
Caching Apple's Signature Server
Tail Call: The New Twitter @Reply
Debian & Android Together on G1
iPhone Theming on WinterBoard
Bypassing iPhone Code Signature
How to Host a Cydia Repository
iPhone Applications in Python
Upgrading the iPhone Toolchain
Bringing Debian APT to the iPhone

On iPhone Development:

files). If you are also choosing to host an AppTapp Installer repository to support legacy clients, you are *strongly* encouraged to make this name match the AppTapp bundle identifier (except all in lower case).

- *Name*: When the package is shown in Cydia's lists, it is convenient to have a prettier name. This field allows you to override this display with an arbitrary string. This field may change often, whereas the "Package" field is fixed for the lifetime of the package.
- *Version*: A package's version indicates two separate values: the version of the software in the package, and the version of the package itself. These version numbers are separated by a hyphen.
- *Architecture*: This describes what system a package is designed for, as .deb files are used on everything from the iPhone to your desktop computer. The correct value for iPhoneOS 1.0.x/1.1.x is "darwin-arm". If you are deploying to iPhoneOS 1.2/2.x you should use "iphoneos-arm".
- *Description*: This field is a little more complicated than the others, as it may use multiple lines. The first line (after the colon) should contain a short description to be displayed on the package lists underneath the name of the package. Optionally, one can choose to replace that description with an arbitrarily long one that will be displayed on the package details screen. Technically the format for this field is quite complicated, but most of that complexity is currently ignored by Cydia: instead Cydia allows you to place arbitrarily awesome HTML in this field. Each line of this extended description *must* begin with a space. I highly disrecommend using this for HTML, however: you should instead use Depiction: for the description in Cydia and use extended descriptions (which will then be ignored by Cydia) for compatibility with command-line clients. I would normally leave this mess undocumented, but this is so different from APT/dpkg that I feel the need to do a full explanation here. Arguably, at some future point, I should redo this field in Cydia to be parsed correctly, so another way of looking at this is that extended descriptions shouldn't be used with Cydia at all.
- *Homepage*: Often, there is more information that a packager wants to provide about a package than can be listed in the description of the package. Cydia supports a "More Info" field on the details screen that shunts users off to a website of the packager's choice.
- *Depiction*: Pretty much the entire interface of Cydia is a webpage, which makes adding features or new functionality remotely very easy. One thing you might want is to be able to display custom links or screenshots with special formatting... just plain something special (even an advertisement) on your package page. This is done with a "depiction", which is a URL that is loaded into an iframe, replacing the Description: and Homepage: links that are normally presnt. For a good example see WinterBoard's package details page in Cydia. For many packagers this has simply *become* their More Information page, which is only used for backwards compatibility. It does not need to be the same, however. You also may consider not having a Homepage: field at all if you include Depiction:.
- *Maintainer*: The person who *built the package* is called the "maintainer". This is the person who will be contacted with issues relating to the package itself. This should be of the form "their name <email@addre.ss>".
- *Author*: In contrast, the person who *wrote the original software* is called the "author". This name will be shown underneath the name of the package on the details screen. The field is in the same format as "Maintainer".
- *Sponsor*: Finally, there might be someone who is simply providing the influence or the cash to make the package happen. This person should be listed here in the form of "Maintainer" except using a resource URI instead of an e-mail address.
- *Section*: Under the "Install" tab in Cydia, packages are listed by "Section". If you would like to encode a space into your section name, use an underscore (Cydia will automatically convert these).

The package is then built by going to the folder containing MyProgram and running dpkg-deb. The result is a .deb file that may be installed and tested on the iPhone. Depending on the version of dpkg-deb you are using, you may receive a number of warnings about "user-defined fields". This is due to the extensions Cydia has added to the control field format and, assuming they look correct, may be safely ignored.

If you are on a Macintosh, before you perform these steps, you need to do one more thing. When tar files (part of the internal structure of a Debian package) are created on Apple systems a number of extra ._* files are created that contain resource fork information. These folders are then installed along with the package and can conflict with other packages (and really shouldn't exist at all). To turn that off, you need to export the following environment variables:

```
export COPYFILE_DISABLE
```

```
export COPY_EXTENDED_ATTRIBUTES_DISABLE
```

```
[root@desktop:~/cydia]# dpkg-deb -b MyProgram
warning, `MyProgram/DEBIAN/control' contains user-defined field `I
warning, `MyProgram/DEBIAN/control' contains user-defined field `I
warning, `MyProgram/DEBIAN/control' contains user-defined field `A
dpkg-deb: building package `com.saurik.myprogram' in `MyProgram.de
dpkg-deb: ignoring 3 warnings about the control file(s)
[root@desktop:~/cydia]# ls -la MyProgram.deb
-rw-r--r-- 1 root root 906 2008-07-01 07:48 MyProgram.deb
[root@desktop:~/cydia]#
```

Obviously, this is just the tip of the iceberg on what you can do with APT. There are a number of different points in the installation process your package can hook into with shell scripts, you can mark files as being "configuration files" (which are intelligently managed and upgraded only when it makes sense), etc.. For more information please scan to the very bottom of this article for the external resources (this article does not discuss these features as 95% of packages do not need such things).

Step 2: Testing the Package

Now that we've constructed our package, it would be prudent to copy it to a phone to test it. As the .deb file is entirely self contained, this is a relatively easy process. On the phone, "dpkg -i" may be used to install it directly from the file. Once installed, it may be managed by Cydia (which will show the package as being from the source "Local/Unknown") or operated on using apt-get.

```
iPhone:~ root# dpkg -i MyProgram.deb
Selecting previously deselected package com.saurik.myprogram.
(Reading database ... 17090 files and directories currently insta
Unpacking com.saurik.myprogram (from MyProgram.deb) ...
Setting up com.saurik.myprogram (1.0.4-1) ...
iPhone:~ root# ls -la /Applications/MyProgram.app
total 0
drwxr-xr-x  2 root wheel  170 Jul  1 00:54 ./
drwxrwxr-x 73 root admin 2550 Jul  1 00:54 ../
-rw-r--r--  1 root wheel    0 Jul  1 00:47 Info.plist
-rw-r--r--  1 root wheel    0 Jul  1 00:47 MyProgram
-rw-r--r--  1 root wheel    0 Jul  1 00:47 icon.png
iPhone:~ root# apt-get remove com.saurik.myprogram
Reading package lists... Done
Building dependency tree... Done
The following packages will be REMOVED:
  com.saurik.myprogram
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
Need to get 0B of archives.
After unpacking 0B of additional disk space will be used.
Do you want to continue [Y/n]? y
(Reading database ... 17095 files and directories currently insta
Removing com.saurik.myprogram ...
iPhone:~ root#
```

Step 3: Creating a Repository

The final step in the process is to create a repository that other users can use to not just install, but *find* your software via Cydia. This step requires creating a folder on a website somewhere which will contain two files: your new .deb and an index called Packages. This index is created using a Perl script called dpkg-scanpackages, which takes the name of a directory to scan for .deb files. Here we pass -m to say "all files, including duplicates" and /dev/null as the "override" file (which will then causes a few warnings which we may ignore).

```
[root@desktop:/web/apt/xmpl]# dpkg-scanpackages -m . /dev/null >P
 ** Packages in archive but missing from override file: **
  com.saurik.myprogram

 Wrote 1 entries to output Packages file.
[root@desktop:/web/apt/xmpl]# bzip2 Packages
```

```
[root@desktop:/web/apt/xmpl]# ls -la *
-rw-r--r-- 1 root root 906 2008-07-01 07:48 MyProgram.deb
-rw-r--r-- 1 root root 380 2008-07-01 08:00 Packages.bz2
[root@desktop:/web/apt/xmpl]#
```

Unfortunately, the stock copy of dpkg-scanpackages that comes with Debian is not designed to take into account user-defined fields, so the Cydia-specific fields will need to be manually added to the script. This is rather easy: simply add "Name", "Author", "Homepage", and "Icon" to the end of the array "fieldpri". If you'd prefer, you may download a pre-made version of this file from my website.

Unfortunately again, this copy I provide will not work with Fink, so you will have to do the rather simple modification yourself. If I get my hands on the stock Fink copy soon I may provide that modified on this website, but currently I don't have it on me :(.

Currently, this program may not be run on the iPhone itself as Perl is not yet ported into Telesphoreo. Hopefully the author will have this fixed in the near future, at which point a few of the more advanced features of dpkg will finally be unlocked (such as dpkg-divert, which allows for safe replacements of files from other packages or the system).

Step 4: Repository Metadata (Optional)

Another feature of a repository is information on the people who run it, what its name is, etc.. This information is stored in a file called Release that has the following format. This file is strictly optional. If you leave it off then Cydia synthesizes information from your URL for various points where it needs to display details on your source.

```
Origin: Saurik's Example for Cydia
Label: Cydia Example
Suite: stable
Version: 0.9
Codename: tangelo
Architectures: iphoneos-arm
Components: main
Description: An Example Repository from HowTo Instructions
```

- *Origin*: This is used by Cydia as the name of the repository as shown in the source editor (and elsewhere). This should be a longer, but not insanely long, description of the repository.
- *Label*: On the package list screens, Cydia shows what repository and section packages came from. This location doesn't have much room, though, so this field should contain a shorter/simpler version of the name of the repository that can be used as a tag.
- *Suite*: Just set this to "stable". This field might not be required, but who really knows? I, for certain, do not.
- *Version*: This is an arbitrary version number that nothing actually parses. I am going to look into seeing how required it is.
- *Codename*: In an "automatic" repository you might store multiple distributions of software for different target systems. For example: apt.saurik.com's main repository houses content both for desktop Debian Etch systems as well as the iPhone. This codename then describes what distribution we are currently looking for. In a "trivial" repository (as described in this document) you may put anything you want here, and the field may even be optional.
- *Architectures*: To verify a repository is for the specific device you are working with APT looks in the release file for this list. You must specify all of the architectures that appear in your Packages file here. Again, we use darwin-arm for 1.1.x and iphoneos-arm for 2.x.
- *Components*: Just set this to "main". This field might not be required, but who really knows? I, for certain, do not.
- *Description*: On the package source screen a short description is listed of the repository. This description may eventually work similarly to that of a package (with a long/short variety and the aforementioned encoding), but for right now only the shorter description is displayed directly on the list.

Step 5: Package Signatures (Optional)

The instructions for Step 5 are not quite done yet. You might just want to skip to Step 6.

The final step, and this is also optional, is to sign your repository. This cryptographic process ensures to your users that the software they are downloading actually comes from

process ensures to your users that the software they are downloading actually comes from you and not from someone on their same network who is intercepting (and manipulating) their traffic. To do this, we first add MD5 sums of our Packages files to Release and then use GnuPG (the GNU version of PGP) to generate a key and sign that file, creating a new Release.gpg file.

For the MD5 sums, we add an extra field call "MD5Sum" which has, on extra lines (beginning with spaces as they are not fields) the sums, size (in bytes), and names of our key files. Note that I believe we have to list *both* the uncompressed and the compressed versions of our files.

```
MD5Sum:
 07a4ca0b91734e0489874dc44bd55222 464 Packages
 43e92f4ec43f4c39d8f4268c7418f353 380 Packages.bz2
```

Somehow, obtain a GPG key... (I don't remember this step).

Now we need to create our Release.gpg file from our Release file. This file will be downloaded by the clients first and then is used to verify the validity of the Release file.

```
[root@desktop:/web/apt/xmpl]# gpg -abs -o Release.gpg Release
You need a passphrase to unlock the secret key for
user: "Jay Freeman (saurik) <saurik@saurik.com>"
1024-bit DSA key, ID AA31C175, created 2008-01-01

Enter passphrase:
[root@desktop:/web/apt/xmpl]# ls -la Release.gpg
-rw-r--r-- 1 root root 189 2008-08-05 11:52 Release.gpg
[root@desktop:/web/apt/xmpl]#
```

In order for this signature to be verified on the client, they have to have our public key installed. This is done by using the program apt-key, a frontend to gpg that handles all of the other work and arguments for us.

```
iPhone:~ root# apt-key add saurik.pub
OK
iPhone:~ root#
```

Step 6: Adding the Repository

Unfortunately, due entirely to the laziness of the author, Cydia does not yet have a nice editor for adding sources. This is lame and will be fixed post haste. In the mean time, to add the repository we just created we need only add a single line to the file /etc/apt/sources.list (which may not exist until you create it) that states the base URL of the entire repository and the offset into this sub-branch (which will usually be "./", but may be more complicated in some cases).

```
deb http://apt.saurik.com/xmpl/ ./
```

This example line is actually a live repository that you may feel free to test or pull apart and analyze based on the instructions in this document. Inside you will find just the two files that we have discussed, and you may install the package containing MyProgram (which doesn't do anything and will probably cause issues if you run it).

Now that we've installed the repository, we may go ahead and test it. To do this, we can either just enter Cydia (which will automatically pick up the new source as it updates its package database) or we can use the command line tool "apt-get". The former is quite obvious, so I will cover the latter here. First we "apt-get update" to pick up the new source information, and then we use "apt-get install" to get our package.

```
iPhone:~ root# apt-get update
Ign http://apt.saurik.com ./ Release.gpg
Ign http://apt.saurik.com ./ Release
Ign http://apt.saurik.com ./ Packages/DiffIndex
Ign http://apt.saurik.com ./ Packages
Get:1 http://apt.saurik.com ./ Packages [349B]
Fetched 33.5kB in 3s (8482B/s)
Reading package lists... Done
iPhone:~ root# apt-get install com.saurik.myprogram
Reading package lists... Done
Building dependency tree... Done
The following NEW packages will be installed:
  com.saurik.myprogram
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

```
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 904B of archives.
After unpacking 0B of additional disk space will be used.
WARNING: The following packages cannot be authenticated!
  com.saurik.myprogram
Install these packages without verification [y/N]? y
Get:1 http://apt.saurik.com ./ com.saurik.myprogram 1.0.4-1 [904B
Fetched 904B in 0s (2515B/s)
Selecting previously deselected package com.saurik.myprogram.
(Reading database ... 17090 files and directories currently insta
Unpacking com.saurik.myprogram (from .../com.saurik.myprogram_1.0
Setting up com.saurik.myprogram (1.0.4-1) ...
iPhone:~ root#
```

More Information

At this point you should be fully ready to head off into the world of distributing iPhone software. However, it should be noted that this is just the tip of the iceberg. What we've now constructed is a "Trivial Repository", and a rather bare-bones one at that. While more than adequate for most usages, some users will have more complicated requirements. For these users I may refer the following resources:

- Debian Repository HOWTO
- Debian Policy Manual
- The Debian System

Finally, in order to help consolidate the resources I've found on {pre,post}{inst,rm} ordering (with all arguments from old and new packages during various operations) I am compiling the following list:

- Random Mailing List Post
- Random Guy's Website

Copyright © 2008–2015 — Jay Freeman (saurik)